IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Broyles et al.**	§	
	§	Group Art Unit: **2181**
Serial No. **10/631,060**	§	
	§	Examiner: **Richard B. Franklin**
Filed: **July 31, 2003**	§	
	§	
For: **Method, System and Product For Serializing Hardware Reset Requests in a Software Communication Request Queue**	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on December 14, 2006.

A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-24.

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: None.
2. Claims withdrawn from consideration but not canceled: None.
3. Claims pending: 1-24
4. Claims allowed: None.
5. Claims rejected: 1-24.
6. Claims objected to: None

C. CLAIMS ON APPEAL

The claims on appeal are: 1-24.

STATUS OF AMENDMENTS

The Final Office Action was mailed October 30, 2006. The Response to Final Office Action was filed December 1, 2006. An Advisory Action Before the Filing of an Appeal Brief was mailed December 11, 2006. A Notice of Appeal was filed December 14, 2006.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of claim 1 is directed to a method in a computer system for serializing hardware reset requests in a software communication request queue in a processor card. (Specification page 6, lines 5-20; and Figure 3.) The processor card processes software requests utilizing the queue in a serial order. (Specification page 6, lines 5-20; and Figure 3.) The computer system includes the processor card and a second resource card. (Specification page 6, lines 5-20; and Figure 3.) The processor card receives a hardware reset request that requests the processor card to reset the second resource card. (Specification page 11, lines 14-19.) The hardware reset request is placed in the queue that is included in the processor card. (Specification page 10, lines 19-22, and page 11, lines 4-19.) Requests from the queue are processed in the serial order, such that the hardware reset request is processed when all requests from the queue that are currently being serviced have completed being serviced. (Specification page 11, lines 20-27, and page 12, lines 5-18.) The processor card resets the second resource card in response to the hardware reset request being processed. (Specification page 13, lines 16-25.)

B. CLAIM 2 – DEPENDENT

Claim 2 depends from claim 1. The computer system includes a plurality of resource cards. The second resource card is included within the plurality of resource cards. (Specification page 10, lines 4-6.) The processor card is coupled to the plurality of resource cards utilizing a single reset bus. (Specification page 10, lines 23-30.) The processor card utilizes the reset line to reset all of the plurality of resource cards simultaneously in response to the receipt of the hardware reset request. (Specification page 10, lines 23-30.)

C. CLAIM 3 - DEPENDENT

Claim 3 depends from claim 2. Software communication requests and hardware reset requests are received. The software communication requests and hardware reset requests are placed in the queue in the serial order in which the software communication requests and

hardware reset requests were received, such that the hardware reset requests are serialized within the queue with the software communication requests. (Specification page 11, lines 4-19.)

D. CLAIM 4 - DEPENDENT

Claim 4 depends from claim 2. A next request in the serial order in the queue is looked at. A determination is made as to whether the next request is a hardware reset request. If it is determined that the next request is a hardware reset request, a determination is made as to whether all of the plurality of resource cards have completed servicing of any pending software communication requests. Execution of the hardware reset request is postponed until all of the plurality of resource cards have completed servicing of any pending software communication requests. (Specification page 11, line 20, through page 12, line 13.)

E. CLAIM 9 - INDEPENDENT

The subject matter of claim 9 is directed to a computer system for serializing hardware reset requests in a software communication request queue in a processor card. (Specification page 6, lines 5-20; and Figure 3.) The processor card processes software requests utilizing the queue in a serial order. (Specification page 6, lines 5-20; and Figure 3.) The computer system includes the processor card and a second resource card. (Specification page 8, line 13, through page 9, line 23.) The processor card receives a hardware reset request that requests the processor card to reset the second resource card. (Specification page 11, lines 14-19.) The queue, which is included in the processor card, stores the hardware reset request. (Specification page 10, lines 19-22, and page 11, lines 4-19) The queue processes requests that are stored in the queue in the serial order such that the hardware reset request is processed from the queue in the serial order when all requests from the queue currently being serviced have completed being serviced. (Specification page 11, lines 20-27, and page 12, lines 5-18.) The processor card resets the second resource card in response to the hardware reset request being processed. (Specification page 13, lines 16-25.)

F. CLAIM 10 - DEPENDENT

Claim 10 depends from claim 9. The computer system includes a plurality of resource cards, which includes the second resource card. (Specification page 10, lines 4-6.) The processor card is coupled to the plurality of resource cards utilizing a single reset bus. (Specification page 10, lines 23-30.) The processor card uses the reset line to reset all of the plurality of resource cards simultaneously in response to the receipt of the hardware reset request. (Specification page 10, lines 23-30.)

G. CLAIM 11 - DEPENDENT

Claim 11 depends from claim 10. The queue stores software communication requests and hardware reset requests in the serial order in which the software communication requests and hardware reset requests were received. The hardware reset requests are serialized within the queue with the software communication requests. (Specification page 11, lines 4-19.)

H. CLAIM 12 – DEPENDENT

Claim 12 depends from claim 10. The system includes a CPU executing code for looking at a next request in the serial order in the queue. The CPU executes code for determining whether the next request is a hardware reset request. If the next request is a hardware reset request, the CPU executes code for determining whether all of the plurality of resource cards have completed servicing of any pending software communication requests. The processor card waits to execute the hardware reset request until all of the plurality of resource cards have completed servicing of any pending software communication requests. (Specification page 8, line 13, through page 9, line 23; and page 11, line 20, through page 12, line 13.)

I. CLAIM 17 - INDEPENDENT

The subject matter of claim 17 is directed to a computer program product that is stored in a storage medium in a computer system for serializing hardware reset requests in a software

communication request queue in a processor card. (Specification page 6, lines 5-20, and page 14, lines 4-16.; and Figure 3.) The processor card processes software requests utilizing the queue in a serial order. (Specification page 6, lines 5-20; and Figure 3.) The computer system includes the processor card and a second resource card. (Specification page 6, lines 5-20; and Figure 3.) First instruction means receive, within the processor card, a hardware reset request that requests the processor card to reset the second resource card. (Specification page 11, lines 14-19.) Second instruction means place the hardware reset request in the queue that is included in the processor card. (Specification page 10, lines 19-22; and page 11, lines 4-19.) Third instruction means process requests from the queue in the serial order such that the hardware reset request is processed when all requests from the queue currently being serviced have completed being serviced. (Specification page 11, lines 20-27, and page 12, lines 5-18.) Fourth instruction means reset, by the processor card, the second resource card in response to the hardware reset request being processed. (Specification page 13, lines 16-25.)

J. CLAIM 18 - DEPENDENT

Claim 18 depends from claim 17. A plurality of resource cards includes the second resource card. (Specification page 10, lines 4-6.) The processor card is coupled to the plurality of resource cards utilizing a single reset bus. (Specification page 10, lines 23-30.) Fifth instruction means reset, by the processor card utilizing the reset line, all of the plurality of resource cards simultaneously in response to the receipt of the hardware reset request. (Specification page 10, lines 23-30.)

K. CLAIM 19 - DEPENDENT

Claim 19 depends from claim 18. Sixth instruction means receive software communication requests and hardware reset requests. Seventh instruction means place the software communication requests and hardware reset requests in the queue in the serial order in which the software communication requests and hardware reset requests were received such that the hardware reset requests are serialized within the queue with said software communication requests. (Specification page 11, lines 4-19.)

L. CLAIM 20 - DEPENDENT

Claim 20 depends from claim 18. Sixth instruction means look at a next request in the serial order in the queue. Seventh instruction means determine whether the next request is a hardware reset request. In response to a determination that the next request is a hardware reset request, eighth instruction means determine whether all of the plurality of resource cards have completed servicing of any pending software communication requests. Ninth instruction means wait to execute the hardware reset request until all of the plurality of resource cards have completed servicing of any pending software communication requests. (Specification page 11, line 20, through page 12, line 13.)

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to review on appeal are as follows:

1. Whether claims 1, 9, and 17 are anticipated by U.S. Patent 6,757,772, issued to *Ono*, under 35 U.S.C. § 102(e); and

2. Whether claims 2-8, 10-16, and 18-24 are obvious over U.S. Patent 6,757,772, issued to *Ono*, in view of Applicants' Admitted Prior Art [hereinafter AAPA] under 35 U.S.C. § 103(a).

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1, 9, and 17)

The Examiner rejected claims 1, 9, and 17 under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent 6,757,772, issued to *Ono*. This position is not well-founded.

Ono does not anticipate Applicants' claims because *Ono* does not teach (1) a queue, or (2) processing requests from said queue in said serial order, said hardware reset request being processed when all requests from said queue currently being service have completed being service.

Ono teaches an interface system that includes multiple nodes that are connected together using a serial interface, such as the IEEE 1394 standard and the USB standard. *Ono*, Figure 1(a), provides an example of such an interface system that includes multiple nodes that are connected together using a bus.

When a node is added to or removed from such an interface system, a bus reset is generated that causes the automatic reassignment of the node addresses of the existing nodes in the interface system. For example, when node F is added to the interface system depicted by Figure 1(a), node E will transmit a bus reset signal to node C, which transmits a bus reset signal to nodes B and D. Node B then transmits a bus reset signal to node A. This results in a bus reset of the entire bus. At this time, the node addresses of nodes A-E are reassigned. See *Ono*, column 4, line 7, through column 8, line 4.

Because a bus reset results in the resetting of the entire bus, data transfers that were occurring during the bus reset are interrupted and must be restarted. Interrupting and restarting some data transfers can be a major hazard or cause a fatal problem. For example, interrupting and restarting print data that is being transferred to a printer should not occur. See *Ono*, column 6, lines 20-46.

Ono attempts to solve this problem by prohibiting a bus reset during specific types of data transfers to specific nodes. When a node, such as node E, detects another node, such as node F, being plugged into node E, node E determines whether data is being transferred from another node to a specific node. If such a transfer is occurring between another node and the specific node, node E will delay issuing a bus reset until the data transfer to the specific node has completed. Once the data transfer to the specific node has been completed, node E will issue a bus reset.

More particularly, *Ono* teaches "When the node plugging or unplugging is detected, the data transfer detector 44 detects if data transfer has occurred relative to the specific node on the bus. First, all data packets that are transferred on the bus are received by the data receiver 441 of the data transmission detector 44 (S102). When the data receiver 441 receives data packets, it transfers all the data packets to the data discriminator 442 regardless of the contents of the transmitting or receiving information such as channel numbers in isochronous transfers or receiving side node addresses in asynchronous transfers. The data discriminator 442 discriminates whether any data related to specific nodes are being transferred judging from data type information or reception/transmission information contained in said data packets transferred by the data receiver 441 (S103)." *Ono*, column 7, lines 5-20.

Using the example depicted in Figure 1(b), node E first receives, through its node plugging/unplugging detector 43, an indication that node F has been plugged into node E. At this time, node E needs to issue a bus reset to the other nodes, i.e. nodes A-D. Instead of immediately issuing the bus reset, however, node E first determines whether a data transfer is currently occurring between another node and a specific node.

As an example, if node C is a computer and node B is a printer, and node C has transferred print data to node B, data discriminator 442 located in node E would detect the transfer of print data to the printer. Node E would then delay issuing a bus reset until the print data transfer is finished. The data that is being transferred did not originate in node E. This is data that is being transferred from node, node C, to a second node, node B.

Applicants claim serializing hardware reset requests in a software communication request queue in a processor card. The processor card processes software requests utilizing the queue in a serial order. The processor card receives a hardware reset request that requests the processor card to reset a second resource card. The hardware reset request is placed in the queue. Requests from the queue are processed in the serial order. The hardware reset request is processed when all requests from the queue currently being serviced have completed being serviced. The processor card resets the second resource card in response to the hardware reset request being processed.

One example of Applicants' use of a queue is described in the Specification, page 11, lines 11-13. "Next, block 404 depicts submitting these software communication requests to a queue. These requests are then stored in the queue." Applicants depict a queue in Figure 3, reference number 322.

Ono does not anticipate Applicants' claims because *Ono* does not teach a queue. Microsoft Computer Dictionary, Fifth Edition, published 2002, defines a queue as "a multi-element data structure from which (by strict definition) elements can be removed only in the same order in which they were inserted; that is, it follows a first in, first out (FIFO) constraint. There are several types of queues in which removal is based on factors other than order of insertion – for example, some priority value assigned to each element." *Ono* does not teach a queue. *Ono* does not teach a data structure.

Ono teaches delaying the issuance of a bus reset. *Ono* does not describe placing the bus reset in a queue. Nowhere in *Ono* is a queue depicted. The process of "delaying" the issuance of a bus reset does not teach a queue.

In the Final Office Action, mailed October 30, 2006, the Examiner stated: "The Examiner submits that delaying a hardware request does teach a queue. The claim does not limit the queue to a physical data structure resident in a computer memory and therefore does not read over a temporal queue. Since a queue is a list with a first-in-first-out structure, delaying a hardware reset request until after completion of a data transfer teaches the structure of a queue. Data transfer requests that were received before the hardware reset request are processed before the execution of the hardware reset. Therefore, delaying the reset request teaches a temporal queue structure." Final Office Action, mailed October 30, 2006, page 3.

Applicants disagree with the Examiner's assertion that the process of "delaying" teaches a queue. As defined, a queue is a data structure. "Delaying" a request is not a data structure.

Further, a queue is defined as being a multi-element data structure from which elements can be removed. Elements cannot be removed from the process of "delaying". Therefore, the process of "delaying" is not a queue.

The Examiner also stated in the Final Office Action on page 3 that a queue is a list. Indeed, using the Examiner's definition of a queue, *Ono* does not teach placing a bus reset in a queue. The bus reset of *Ono* is not placed in a list. *Ono* teaches merely that the data discriminator in a first node waits until the data transfer between two other nodes has finished before the first node issues the bus reset. Thus, the bus reset is not placed in a list.

Because *Ono* does not teach a queue, *Ono* does not teach a queue in a processor card. The Examiner does not cite any section of *Ono* that supposedly teaches a queue in a processor card.

Therefore, because *Ono* does not teach a queue, *Ono* does not anticipate Applicants' claims.

Applicants also claim processing requests from said queue in said serial order, said

hardware reset request being processed when all requests from said queue currently being serviced have completed being serviced. *Ono* does not teach processing a hardware request from a queue when all requests from that same queue currently being serviced have completed being serviced.

As described above, the data discriminator in a node, such as node E, discriminates data transfers. These are data transfers on the bus between nodes A-D. These are not data transfers between node E and another node. Thus, node E is, in effect, snooping data transfers on the bus. The data discriminator in node E receives data packets that are being transferred on the bus from one node to another. This is not data that node E is transmitting or receiving.

According to Applicants' claims, the hardware reset request and other requests are processed from the same queue. According to *Ono*, a first node's bus reset signal is processed from the first node, while the data transfer is processed between two other nodes. The bus reset signal of *Ono* and the data transfer that is causing the bus reset signal to be delayed are not processed from the same queue.

The Examiner does not refer to any section of *Ono* that teaches the bus reset signal and the data transfer being processed from the same queue.

Because *Ono* does not teach processing requests from said queue in said serial order, said hardware reset request being processed when all requests from said queue currently being serviced have completed being serviced, *Ono* does not anticipate Applicants' claims.

B. GROUND OF REJECTION 2 (Claims 2-8, 10-16, and 18-24)

The Examiner rejected claims 2-8, 10-16, and 18-24 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent 6,757,772, issued to *Ono*, in view of Applicants Admitted Prior Art [hereinafter AAPA]. This position is not well-founded.

B.1. Claims 2, 10, and 18

Applicants' claim 2 recites: The method according to claim 1, further comprising the steps of: said computer system including a plurality of resource cards, said second resource card included within said plurality of resource cards; said processor card coupled to said plurality of resource cards utilizing a single reset bus; and resetting, by said processor card utilizing said reset

line, all of said plurality of resource cards simultaneously in response to said receipt of said hardware reset request. Applicants' claims 10 and 18 recite similar features.

The Examiner states that *Ono* does not teach said processor card coupled to said plurality of resource cards utilizing a single reset bus; and resetting, by said processor card utilizing said reset line, all of said plurality of resource cards simultaneously in response to said receipt of said hardware reset request. The Examiner relies on AAPA to cure the deficiencies of *Ono*.

Applicants' Admitted Prior Art [AAPA] does not cure the deficiencies of *Ono* because AAPA does not teach receiving, within said processor card, a hardware reset request that requests said processor card to reset said second resource card; placing said hardware reset request in said queue that is included in said processor card; processing requests from said queue in said serial order, said hardware reset request being processed when all requests from said queue currently being serviced have completed being serviced; and resetting, by said processor card, said second resource card in response to said hardware reset request being processed. Therefore, the combination of *Ono* and AAPA does not render Applicants' claims 2, 10, and 18 obvious.

B.2. Claims 3, 11, and 19

Applicants' claim 3 recites: The method according to claim 2, further comprising the steps of: receiving software communication requests and hardware reset requests; placing said software communication requests and hardware reset requests in said queue in said serial order in which said software communication requests and hardware reset requests were received, said hardware reset requests being serialized within said queue with said software communication requests. Applicants' claims 11 and 19 recite similar features.

The Examiner states that AAPA teaches placing software communication requests in a queue, and *Ono* teaches delaying the hardware reset requests until all software communication requests have completed. The Examiner does not assert that *Ono* teaches placing hardware reset requests in a queue. The Examiner does not assert that the combination of *Ono* and AAPA teaches placing said software communication requests and hardware reset requests in said queue in said serial order. The Examiner does not assert that the combination of *Ono* and AAPA teaches said hardware reset requests being serialized within said queue with said software communication requests.

The combination of *Ono* and AAPA does not teach placing said software communication requests and hardware reset requests in said queue in said serial order in which said software communication requests and hardware reset requests were received, said hardware reset requests being serialized within said queue with said software communication requests. Therefore, the combination of *Ono* and AAPA does not render Applicants' claims 3, 11, and 19 obvious.

B.3. Claims 4, 12, and 20

Applicants' claim 4 recites: The method according to claim 2, further comprising the steps of: looking at a next request in said serial order in said queue; determining whether said next request is a hardware reset request; in response to a determination that said next request is a hardware reset request, determining whether all of said plurality of resource cards have completed servicing of any pending software communication requests; and waiting to execute said hardware reset request until all of said plurality of resource cards have completed servicing of any pending software communication requests. Applicants' claims 12 and 20 recite similar features.

The Examiner does not assert that the combination of *Ono* and AAPA teaches looking at a next request in said serial order in said queue.

The combination of *Ono* and AAPA does not teach the features of claims 4, 12, and 20 because the combination does not teach a queue. Because the combination does not teach a queue, the combination cannot teach looking at a next request in said serial order in said queue. Therefore, the combination of *Ono* and AAPA does not render Applicants' claims 4, 12, and 20 obvious.

Claims 13-16 and 21-24 depend from the independent claims discussed above and are patentable for the reasons given above.

C. Conclusion

Ono does not anticipate Applicants' claims 1, 9, and 17 because *Ono* does not teach (1) a queue, or (2) processing requests from said queue in said serial order, said hardware reset request being processed when all requests from said queue currently being service have completed being service.

The combination of *Ono* and Applicants' Admitted Prior Art does not render Applicants' claims 2-8, 10-16, and 18-24 obvious because the combination does not teach (1) a queue, or (2) processing requests from said queue in said serial order, said hardware reset request being processed when all requests from said queue currently being service have completed being service in combination with the features of these claims.

/Lisa L. B. Yociss/
Lisa L. B. Yociss
Reg. No. 36,975
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

1. A method in a computer system for serializing hardware reset requests in a software communication request queue in a processor card, said processor card processing software requests utilizing said queue in a serial order, said computer system including said processor card and a second resource card, said method comprising the steps of:

receiving, within said processor card, a hardware reset request that requests said processor card to reset said second resource card;

placing said hardware reset request in said queue that is included in said processor card;

processing requests from said queue in said serial order, said hardware reset request being processed when all requests from said queue currently being serviced have completed being serviced; and

resetting, by said processor card, said second resource card in response to said hardware reset request being processed.

2. The method according to claim 1, further comprising the steps of:

said computer system including a plurality of resource cards, said second resource card included within said plurality of resource cards;

said processor card coupled to said plurality of resource cards utilizing a single reset bus; and

resetting, by said processor card utilizing said reset line, all of said plurality of resource cards simultaneously in response to said receipt of said hardware reset request.

3. The method according to claim 2, further comprising the steps of:

receiving software communication requests and hardware reset requests;

placing said software communication requests and hardware reset requests in said queue in said serial order in which said software communication requests and hardware reset requests were received, said hardware reset requests being serialized within said queue with said software communication requests.

4. The method according to claim 2, further comprising the steps of:

looking at a next request in said serial order in said queue;

determining whether said next request is a hardware reset request;

in response to a determination that said next request is a hardware reset request, determining whether all of said plurality of resource cards have completed servicing of any pending software communication requests; and

waiting to execute said hardware reset request until all of said plurality of resource cards have completed servicing of any pending software communication requests.

5. The method according to claim 4, further comprising the steps of:

in response to determining that all of said plurality of resource cards have completed servicing of any pending software communication requests, determining whether all of said plurality of resource cards have completed servicing of any pending sequence of multiple software communication requests; and

waiting to execute said hardware reset request until all of said plurality of resource cards have completed servicing of any pending sequence of multiple software communication requests.

6. The method according to claim 1, further comprising the steps of:

said processor card coupled to said second resource card utilizing a hardware reset line; and

resetting, by said processor card, said second resource card by pulling said hardware reset line high.

7. The method according to claim 1, further comprising the steps of:

said second resource card including a microcontroller;

said hardware reset request requesting said processor card to reset said microcontroller; and

resetting, by said processor card, said microcontroller in said second resource card in response to said hardware reset request being processed.

8. The method according to claim 2, further comprising the steps of:

each one of said plurality of resource cards including a microcontroller, a memory, and synchronization bits;

utilizing said synchronization bits to maintain information about current servicing of software communication requests by each one of said plurality of resource cards;

resetting said microcontroller and said synchronization bits in each one of said plurality of resource cards simultaneously in response to said receipt said hardware reset request specifying one of a plurality of resource cards to reset; and

losing said information about current servicing of software communication requests by each one of said plurality of resource cards when said synchronization bits are reset.

9. A computer system for serializing hardware reset requests in a software communication request queue in a processor card, said processor card processing software requests utilizing said queue in a serial order, said computer system including said processor card and a second resource card, said system comprising:

said processor card receiving a hardware reset request that requests said processor card to reset said second resource card;

said queue, which is included in said processor card, for storing said hardware reset request;

said queue for processing requests that are stored in said queue in said serial order, said hardware reset request being processed from said queue in said serial order when all requests from said queue currently being serviced have completed being serviced; and

said processor card resetting said second resource card in response to said hardware reset request being processed.

10. The system according to claim 9, further comprising:

said computer system including a plurality of resource cards, said second resource card included within said plurality of resource cards;

said processor card coupled to said plurality of resource cards utilizing a single reset bus; and

said processor card using said reset line to reset all of said plurality of resource cards simultaneously in response to said receipt of said hardware reset request.

11. The system according to claim 10, further comprising:

said queue for storing software communication requests and hardware reset requests in said serial order in which said software communication requests and hardware reset requests were received, said hardware reset requests being serialized within said queue with said software communication requests.

12. The system according to claim 10, further comprising:

said system including a CPU executing code for looking at a next request in said serial order in said queue;

said CPU executing code for determining whether said next request is a hardware reset request;

in response to a determination that said next request is a hardware reset request, said CPU executing code for determining whether all of said plurality of resource cards have completed servicing of any pending software communication requests; and

said processor card for waiting to execute said hardware reset request until all of said plurality of resource cards have completed servicing of any pending software communication requests.

13. The system according to claim 12, further comprising:

in response to determining that all of said plurality of resource cards have completed servicing of any pending software communication requests, said CPU executing code for determining whether all of said plurality of resource cards have completed servicing of any pending sequence of multiple software communication requests; and

said processor card for waiting to execute said hardware reset request until all of said plurality of resource cards have completed servicing of any pending sequence of multiple software communication requests.

14. The system according to claim 9, further comprising:

said processor card coupled to said second resource card utilizing a hardware reset line; and

said processor card resetting said second resource card by pulling said hardware reset line high.

15. The system according to claim 9, further comprising:

said second resource card including a microcontroller;

said hardware reset request requesting said processor card to reset said microcontroller; and

said processor card resetting said microcontroller in said second resource card in response to said hardware reset request being processed.

16. The system according to claim 10, further comprising:

each one of said plurality of resource cards including a microcontroller, a memory, and synchronization bits;

said synchronization bits for maintaining information about current servicing of software communication requests by each one of said plurality of resource cards;

said microcontroller and said synchronization bits being reset in each one of said plurality of resource cards simultaneously in response to said receipt said hardware reset request

specifying one of a plurality of resource cards to reset; and

said information about current servicing of software communication requests being lost by each one of said plurality of resource cards when said synchronization bits are reset.

17. A computer program product that is stored in a storage medium in a computer system for serializing hardware reset requests in a software communication request queue in a processor card, said processor card processing software requests utilizing said queue in a serial order, said computer system including said processor card and a second resource card, said product comprising:

first instruction means for receiving, within said processor card, a hardware reset request that requests said processor card to reset said second resource card;

second instruction means for placing said hardware reset request in said queue that is included in said processor card;

third instruction means for processing requests from said queue in said serial order, said hardware reset request being processed when all requests from said queue currently being serviced have completed being serviced; and

fourth instruction means for resetting, by said processor card, said second resource card in response to said hardware reset request being processed.

18. The product according to claim 17, further comprising:

said computer system including a plurality of resource cards, said second resource card included within said plurality of resource cards;

said processor card coupled to said plurality of resource cards utilizing a single reset bus; and

fifth instruction means for resetting, by said processor card utilizing said reset line, all of said plurality of resource cards simultaneously in response to said receipt of said hardware reset request.

19. The product according to claim 18, further comprising:

sixth instruction means for receiving software communication requests and hardware reset requests;

seventh instruction means for placing said software communication requests and hardware reset requests in said queue in said serial order in which said software communication requests and hardware reset requests were received, said hardware reset requests being serialized within said queue with said software communication requests.

20. The product according to claim 18, further comprising:

sixth instruction means for looking at a next request in said serial order in said queue;

seventh instruction means for determining whether said next request is a hardware reset request;

in response to a determination that said next request is a hardware reset request, eighth instruction means for determining whether all of said plurality of resource cards have completed servicing of any pending software communication requests; and

ninth instruction means for waiting to execute said hardware reset request until all of said plurality of resource cards have completed servicing of any pending software communication requests.

21. The product according to claim 20, further comprising:

in response to determining that all of said plurality of resource cards have completed servicing of any pending software communication requests, tenth instruction means for determining whether all of said plurality of resource cards have completed servicing of any pending sequence of multiple software communication requests; and

eleventh instruction means for waiting to execute said hardware reset request until all of said plurality of resource cards have completed servicing of any pending sequence of multiple software communication requests.

22. The product according to claim 17, further comprising:

said processor card coupled to said second resource card utilizing a hardware reset line; and

fifth instruction means for resetting, by said processor card, said second resource card by pulling said hardware reset line high.

23. The product according to claim 17, further comprising:

said second resource card including a microcontroller;

said hardware reset request requesting said processor card to reset said microcontroller; and

fifth instruction means for resetting, by said processor card, said microcontroller in said second resource card in response to said hardware reset request being processed.

24. The product according to claim 18, further comprising:

each one of said plurality of resource cards including a microcontroller, a memory, and synchronization bits;

sixth instruction means for utilizing said synchronization bits to maintain information about current servicing of software communication requests by each one of said plurality of resource cards;

seventh instruction means for resetting said microcontroller and said synchronization bits in each one of said plurality of resource cards simultaneously in response to said receipt said hardware reset request specifying one of a plurality of resource cards to reset; and

eighth instruction means for losing said information about current servicing of software communication requests by each one of said plurality of resource cards when said synchronization bits are reset.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.